SimpliLearn Capstone Project 1 : Real Estate

---Lily

Real Estate.

Course-end Project 1

DESCRIPTION

Problem Statement

A banking institution requires actionable insights into mortgage-backed securities, geographic business investment, and real estate analysis. The mortgage bank would like to identify potential monthly mortgage expenses for each region based on monthly family income and rental of the real estate. A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region in the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies. The dashboard must demonstrate relationships and trends for the key metrics as follows: number of loans, average rental income, monthly mortgage and owner's cost, family income vs mortgage cost comparison across different regions. The metrics described here do not limit the dashboard to these few. Dataset Description

Variables

Description Second mortgage Households with a second mortgage statistics Home equity Households with a home equity loan statistics Debt Households with any type of debt statistics Mortgage Costs Statistics regarding mortgage payments, home equity loans, utilities, and property taxes Home Owner Costs Sum of utilities, and property taxes statistics Gross Rent Contract rent plus the estimated average monthly cost of utility features High school Graduation High school graduation statistics Population Demographics Population demographics statistics Age Demographics Age demographic statistics Household Income Total income of people residing in the household Family Income Total income of people related to the householder

Project Task: Week 1

Data Import and Preparation:

1. Import data.
2. Figure out the primary key and look for the requirement of indexing.

3. Gauge the fill rate of the variables and devise plans for missing value treatment. Please explain explicitly the reason for the treatment chosen for each variable.
4. Perform debt analysis. You may take the following steps:

Exploratory Data Analysis (EDA):

a) Explore the top 2,500 locations where the percentage of households with a second mortgage is the highest and percent ownership is above 10 percent. Visualize using geo-map. You may keep the upper limit for the percent of households with a second mortgage to 50 percent

b) Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage c) Create pie charts to show overall debt and bad debt

d) Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities

e) Create a collated income distribution chart for family income, house hold income, and remaining income

Project Task: Week 2

Exploratory Data Analysis (EDA):

1. Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):

a) Use pop and ALand variables to create a new field called population density

b) Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age c) Visualize the findings using appropriate chart type

2. Create bins for population into a new variable by selecting appropriate class interval so that the number of categories don't exceed 5 for the ease of analysis.

a) Analyze the married, separated, and divorced population for these population brackets

b) Visualize using appropriate chart type

3. Please detail your observations for rent as a percentage of income at an overall level, and for different states.
4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

Project Task: Week 3

Data Pre-processing:

1. The economic multivariate data has a significant number of measured variables. The goal is to find where the measured variables depend on a number of smaller unobserved common factors or latent variables. 2. Each variable is assumed to be dependent upon a linear combination of the common factors, and the coefficients are known as loadings. Each measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable. Obtain the common factors and then plot the loadings. Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships in the data. Following are the list of latent variables:

• Highschool graduation rates

• Median population age

• Second mortgage statistics

• Percent own

• Bad debt expense

Project Task: Week 4

Data Modeling :

1. Build a linear Regression model to predict the total monthly expenditure for home mortgages loan. Please refer ‘deplotment_RE.xlsx’. Column hc_mortgage_mean is predicted variable. This is the mean monthly mortgage and owner costs of specified geographical location. Note: Exclude loans from prediction model which have NaN (Not a Number) values for hc_mortgage_mean.

a) Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step.

b) Run another model at State level. There are 52 states in USA.

c) Keep below considerations while building a linear regression model. Data Modeling :

• Variables should have significant impact on predicting Monthly mortgage and owner costs

• Utilize all predictor variable to start with initial hypothesis

• R square of 60 percent and above should be achieved

• Ensure Multi-collinearity does not exist in dependent variables

• Test if predicted variable is normally distributed

Data Reporting:

2. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

a) Box plot of distribution of average rent by type of place (village, urban, town, etc.).

b) Pie charts to show overall debt and bad debt.

c) Explore the top 2,500 locations where the percentage of households with a second mortgage is the highest and percent ownership is above 10 percent. Visualize using geo-map.

d) Heat map for correlation matrix.

e) Pie chart to show the population distribution across different types of places (village, urban, town etc.)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
%matplotlib inline
```

WEEK 1:

1. Import data.

```
#import both the datasets

df_train=pd.read_csv('train.csv')
df_test=pd.read_csv('test.csv')
```

```
df_train.shape
```

```
(27321, 80)
```

```
df_test.shape
```

```
(11709, 80)
```

```
#check the train dataset

df_train.head()
```

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City

5 rows × 80 columns

```
#check for null values for train data

df_train.isnull().sum()
```

```
UID               0
BLOCKID       27321
SUMLEVEL          0
COUNTYID          0
STATEID           0
              ...
pct_own         268
married         191
married_snp     191
separated       191
divorced        191
Length: 80, dtype: int64
```

```
#check the test dataset

df_test.head()
```

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	pla
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearb Heig (
1	252676	NaN	140	1	23	Maine	ME	Auburn	Aub (
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Miller
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Montic (
4	286865	NaN	140	355	48	Texas	TX	Corpus	Ed

```
#check the test data info and null values

df_test.info()
```

```
 24  rent_samples               11561 non-null  float64
 25  rent_gt_10                 11560 non-null  float64
 26  rent_gt_15                 11560 non-null  float64
 27  rent_gt_20                 11560 non-null  float64
 28  rent_gt_25                 11560 non-null  float64
 29  rent_gt_30                 11560 non-null  float64
 30  rent_gt_35                 11560 non-null  float64
```

```
 60  home_equity_cdf              11489 non-null  float64
 61  debt_cdf                     11489 non-null  float64
 62  hs_degree                    11624 non-null  float64
 63  hs_degree_male               11620 non-null  float64
 64  hs_degree_female             11604 non-null  float64
 65  male_age_mean                11625 non-null  float64
 66  male_age_median              11625 non-null  float64
 67  male_age_stdev               11625 non-null  float64
 68  male_age_sample_weight       11625 non-null  float64
 69  male_age_samples             11625 non-null  float64
 70  female_age_mean              11613 non-null  float64
 71  female_age_median            11613 non-null  float64
 72  female_age_stdev             11613 non-null  float64
 73  female_age_sample_weight     11613 non-null  float64
 74  female_age_samples           11613 non-null  float64
 75  pct_own                      11587 non-null  float64
 76  married                      11625 non-null  float64
 77  married_snp                  11625 non-null  float64
 78  separated                    11625 non-null  float64
 79  divorced                     11625 non-null  float64
dtypes: float64(61), int64(13), object(6)
memory usage: 7.1+ MB
```

```
df_test.isnull().sum()
```

```
UID               0
BLOCKID       11709
SUMLEVEL          0
COUNTYID          0
STATEID           0
                ...
pct_own         122
married          84
married_snp      84
separated        84
divorced         84
Length: 80, dtype: int64
```

```
#total rows in train data

len(df_train)
```

```
27321
```

```
#total null values in column "BLOCKID" from train data

df_train['BLOCKID'].isnull().sum()
```

```
27321
```

```
#total rows in test data
```

```
len(df_test)
```

```
11709
```

```
#total null values in column "BLOCKID" from test data

df_test['BLOCKID'].isnull().sum()
```

```
11709
```

From above two cells it is clear that

we have no data in the BLOCK ID column

So we can drop this column

```
# delete BLOCKEDID column from train data

df_train=df_train.drop('BLOCKID',axis=1)
```

```
# delete BLOCKEDID column from test data

df_test=df_test.drop('BLOCKID',axis=1)
```

2. Figure out the primary key and look for the requirement of indexing.

```
#caculating unique value fron train dataset

df_train.nunique()
```

```
UID            27161
SUMLEVEL           1
COUNTYID         296
STATEID           52
state             52
                 ...
pct_own        22302
married        20282
married_snp    10350
separated       6190
divorced       13688
Length: 79, dtype: int64
```

Looking at our output we can conclude that UID column has most unique **value** so we can pick this as primary key

```
#caculating unique value fron test dataset

df_test.nunique()
```

```
UID            11677
SUMLEVEL           1
COUNTYID         246
STATEID           52
state             52
                 ...
pct_own        10578
married        10215
married_snp     6829
separated       4512
divorced        8273
Length: 79, dtype: int64
```

Looking at our output we can conclude that UID column has most unique value so we can pick this as primary key

```
# Checking duplicate values From Train dataset

duplicate_train = df_train[df_train.duplicated()].count()


duplicate_train
```

```
UID            160
SUMLEVEL       160
COUNTYID       160
STATEID        160
state          160
               ...
pct_own         99
married        119
married_snp    119
separated      119
divorced       119
Length: 79, dtype: int64
```

In UID column we can see that we have 160 duplicate column

we should drop these column to avoid overfitting

```
# Checking duplicate values From Test dataset
```

```
duplicate_test = df_test[df_test.duplicated()].count()
```

```
duplicate_test
```

```
UID            32
SUMLEVEL       32
COUNTYID       32
STATEID        32
state          32
               ..
pct_own        22
married        25
married_snp    25
separated      25
divorced       25
Length: 79, dtype: int64
```

- In UID column we can see that we have 32 duplicate column

  we should drop these column to avoid overfitting

```
#deleting duplicate values from train dataset

df_train.drop_duplicates(keep='first',inplace=True)
```

```
#deleting duplicate values from train dataset

df_test.drop_duplicates(keep='first',inplace=True)
```

```
# setting UID as index column in Train dataset
# and using key word setting it as primary key

df_train.set_index(keys='UID',inplace=True)
```

```
df_train.head()
```

UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	prim
267822	140	53	36	New York	NY	Hamilton	Hamilton	City	t
246444	140	141	18	Indiana	IN	South Bend	Roseland	City	t

```
# setting UID as index column in Test dataset
# and using key word setting it as primary key

df_test.set_index(keys='UID',inplace=True)
```

247218 140 161 20 Kansas KS Manhattan City City t

```
df_test.head()
```

UID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type
255504	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP
252676	140	1	23	Maine	ME	Auburn	Auburn City	City
276314	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough
248614	140	231	21	Kentucky	KY	Monticello	Monticello City	City
286865	140	355	48	Texas	TX	Corpus Christi	Edroy	Town

5 rows × 78 columns

Now we can see the no.of column is reduced from 80 to 78. So we can conclude that we have successfully droped the BLOCKID column and set UID as index

3. Gauge the fill rate of the variables and devise plans for missing value treatment. Please explain explicitly the reason for the treatment chosen for each variable.

```
# check for the missing values in train dataset

df_train.isnull().sum()
```

```
SUMLEVEL         0
COUNTYID         0
STATEID          0
state            0
state_ab         0
                ...
pct_own        207
married        150
married_snp    150
separated      150
divorced       150
Length: 78, dtype: int64
```

```
# check for the missing values in test dataset

df_test.isnull().sum()
```

```
SUMLEVEL         0
COUNTYID         0
STATEID          0
state            0
state_ab         0
                ...
pct_own        112
married         77
married_snp     77
separated       77
divorced        77
Length: 78, dtype: int64
```

From above we can say that we have missing values in our train and test dataset.

```
#calculating percentage of missing values from Train dataset and create a dataframe for missi

missing_value_train=df_train.isnull().sum()*100/len(df_train)
df_missing_value_train=pd.DataFrame(missing_value_train,columns=['Percentage of missing value


df_missing_value_train
```

	Percentage of missing value
SUMLEVEL	0.00
COUNTYID	0.00
STATEID	0.00
state	0.00
state_ab	0.00
...	...
pct_own	0.76
married	0.55

```
#sorting in descending order to check highest missing values

df_missing_value_train.sort_values(by=['Percentage of missing value'],ascending=False)
```

	Percentage of missing value
hc_stdev	1.76
hc_mean	1.76
hc_sample_weight	1.76
hc_samples	1.76
hc_median	1.76
...	...
pop	0.00
male_pop	0.00
female_pop	0.00
universe_samples	0.00
SUMLEVEL	0.00

78 rows × 1 columns

```
#calculating percentage of missing values from Test dataset and create a dataframe for missir


missing_value_test=df_test.isnull().sum()*100/len(df_test)
df_missing_value_test=pd.DataFrame(missing_value_test,columns=['Percentage of missing value']


df_missing_value_test
```

	Percentage of missing value
SUMLEVEL	0.00
COUNTYID	0.00
STATEID	0.00
state	0.00
state_ab	0.00
...	...
pct_own	0.96
married	0.66
married_snp	0.66
separated	0.66
divorced	0.66

78 rows × 1 columns

```
#sorting in descending order to check highest missing values

df_missing_value_test.sort_values(by=['Percentage of missing value'],ascending=False)
```

	Percentage of missing value
hc_sample_weight	2.29
hc_samples	2.29
hc_stdev	2.29
hc_median	2.29
hc_mean	2.29
...	...
pop	0.00
male_pop	0.00
female_pop	0.00
universe_samples	0.00
SUMLEVEL	0.00

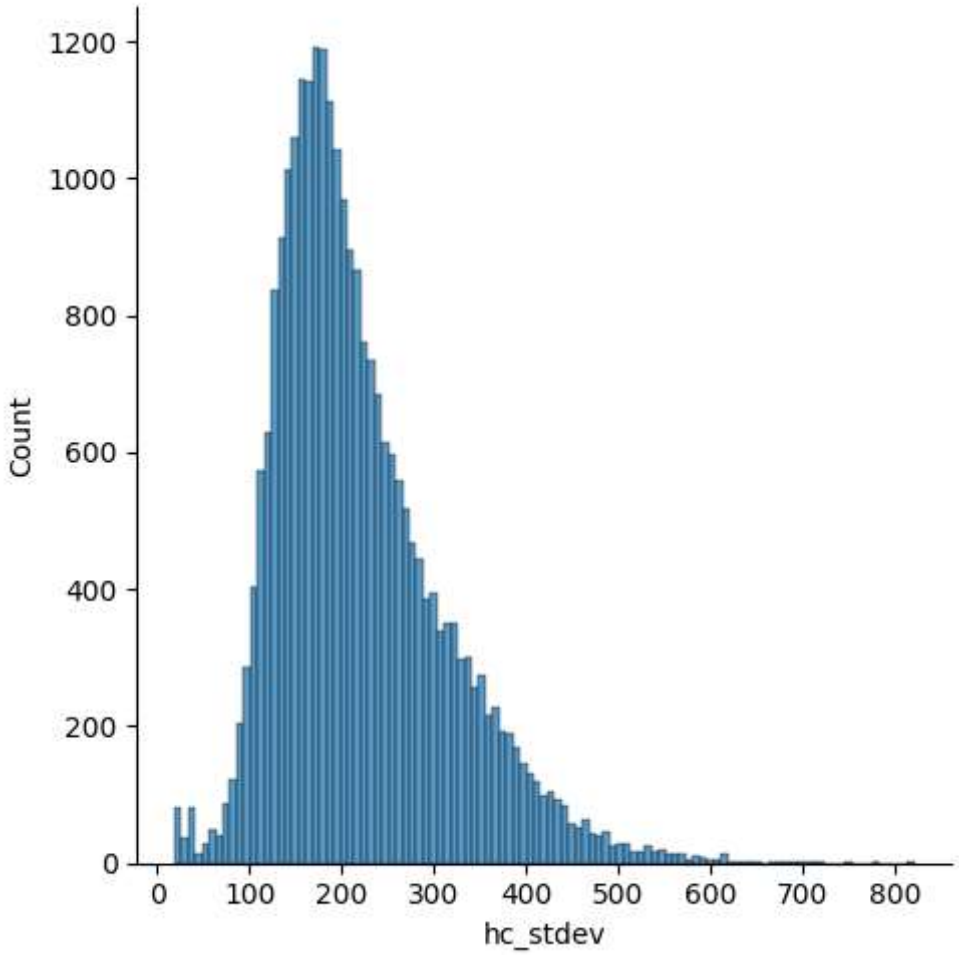
78 rows × 1 columns

from above we observe that the missing value for each column is less than 3% so we will not drop the data but rather fill the data with mean, median or mode.

to check which measure of central tendency is best suited we will check the data for skewness and outliers.

```
# select any column from the dataset and plot the distribution
sns.displot(df_train['hc_stdev'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc05a965850>
```



We can see that our dataset is right skewed so we should prefer median or mode over mean as mean is prone to skewness. and we know that mode is generally used for categorical data so we will use median for filling the missing values.

```
#filling missing data with median in train dataset

df_train.fillna(df_train.median(),inplace=True)
```

```
<ipython-input-797-9b905a3ae00f>:3: FutureWarning: Dropping of nuisance columns in Data
  df_train.fillna(df_train.median(),inplace=True)
```

```
#filling missing data with median in test dataset

df_test.fillna(df_test.median(),inplace=True)
```

```
<ipython-input-798-8d3c22968141>:3: FutureWarning: Dropping of nuisance columns in Data
  df_test.fillna(df_test.median(),inplace=True)
```

Exploratory Data Analysis (EDA):

4. Perform debt analysis. You may take the following steps:

a) Explore the top 2,500 locations where the percentage of households with a second mortgage is the highest and percent ownership is above 10 percent. Visualize using geo-map. You may keep the upper limit for the percent of households with a second mortgage to 50 percent

b) Use the following bad debt equation: Bad Debt = P (Second Mortgage ∩ Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage

```
#Introducing a  new column for Bad Debt in the train dataframe.

df_train['Bad_Debt']=df_train['second_mortgage']+df_train['home_equity']-df_train['home_equit
```

```
#checking values

df_train['Bad_Debt']
```

```
UID
267822    0.09
246444    0.04
245683    0.10
279653    0.01
247218    0.05
          ...
279212    0.00
277856    0.21
233000    0.08
287425    0.14
```

```
265371    0.18
Name: Bad_Debt, Length: 27161, dtype: float64
```

```
#Introducing a  new column for Bad Debt in the train dataframe.

df_test['Bad_Debt']=df_test['second_mortgage']+df_test['home_equity']-df_test['home_equity_se
```

```
#checking values

df_test['Bad_Debt']
```

```
UID
255504    0.08
252676    0.14
276314    0.07
248614    0.02
286865    0.03
          ...
238088    0.06
242811    0.08
250127    0.14
241096    0.08
287763    0.05
Name: Bad_Debt, Length: 11677, dtype: float64
```

c) Create pie charts to show overall debt and bad debt

```
  # creating dataframe for debt and Bad debt

dataframe = pd.DataFrame({'Name': ['Debt', 'Bad_Debt'],
                          'sum': [df_train['Bad_Debt'].sum(), df_train['debt'].sum()]})

# Plotting the pie chart for above dataframe
dataframe.groupby(['Name']).sum().plot(kind='pie', y='sum')
```

```
<Axes: ylabel='sum'>
```

The sum of debt is much less compared to the sum of bad debt.

d) Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities

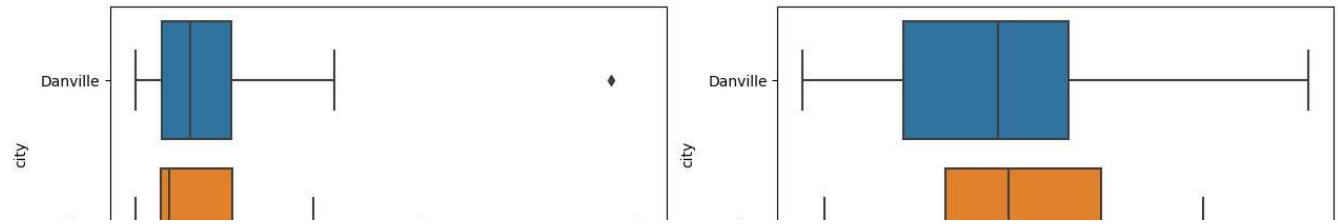
```
# create list of given column

column=['second_mortgage','home_equity','debt','Bad_Debt']


# take any two cities to show plot
# we will take Hamilton and Danville
# and create seprate df for each city

df_Danville=df_train.loc[df_train['city']=='Danville']
df_Hamilton=df_train.loc[df_train['city']=='Hamilton']


# Now concatenate both the datasets

df_city=pd.concat([df_Danville,df_Hamilton])


# Now check the dataset

df_city.head(3)
```

```
              SUMLEVEL  COUNTYID  STATEID    state  state_ab    city   place  type  primary
         UID
```

```
#check total cities value

df_city['city'].value_counts()
```

```
     Danville    28
     Hamilton    19
     Name: city, dtype: int64
     3 rows × 79 columns
```

```
# create box plot using subplots of all given columns =['second_mortgage','home_equity','debt
plt.figure(figsize=(15,8))
plt.subplot(2,·2,·1)
sns.boxplot(data=df_city,x='second_mortgage', y='city')
plt.subplot(2, 2, 2)
sns.boxplot(data=df_city,x='home_equity', y='city')
plt.subplot(2, 2, 3)
sns.boxplot(data=df_city,x='debt', y='city')
plt.subplot(2, 2, 4)
sns.boxplot(data=df_city,x='Bad_Debt', y='city')
plt.show()
```

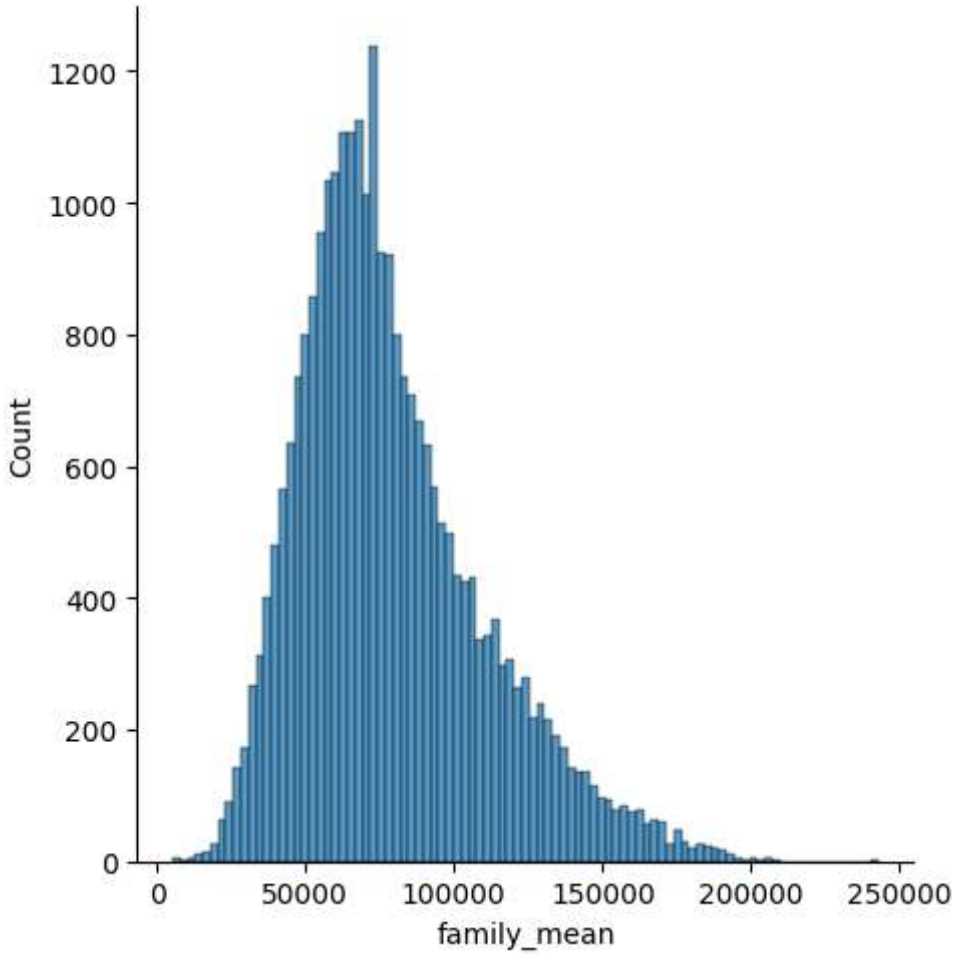


e) Create a collated income distribution chart for family income, house hold income, and remaining income

```
col=['family_mean','hi_mean','rem_incom']
```


```
#Creating DISPLOT for FAMILY INCOME

sns.displot(df_train, x="family_mean")
```

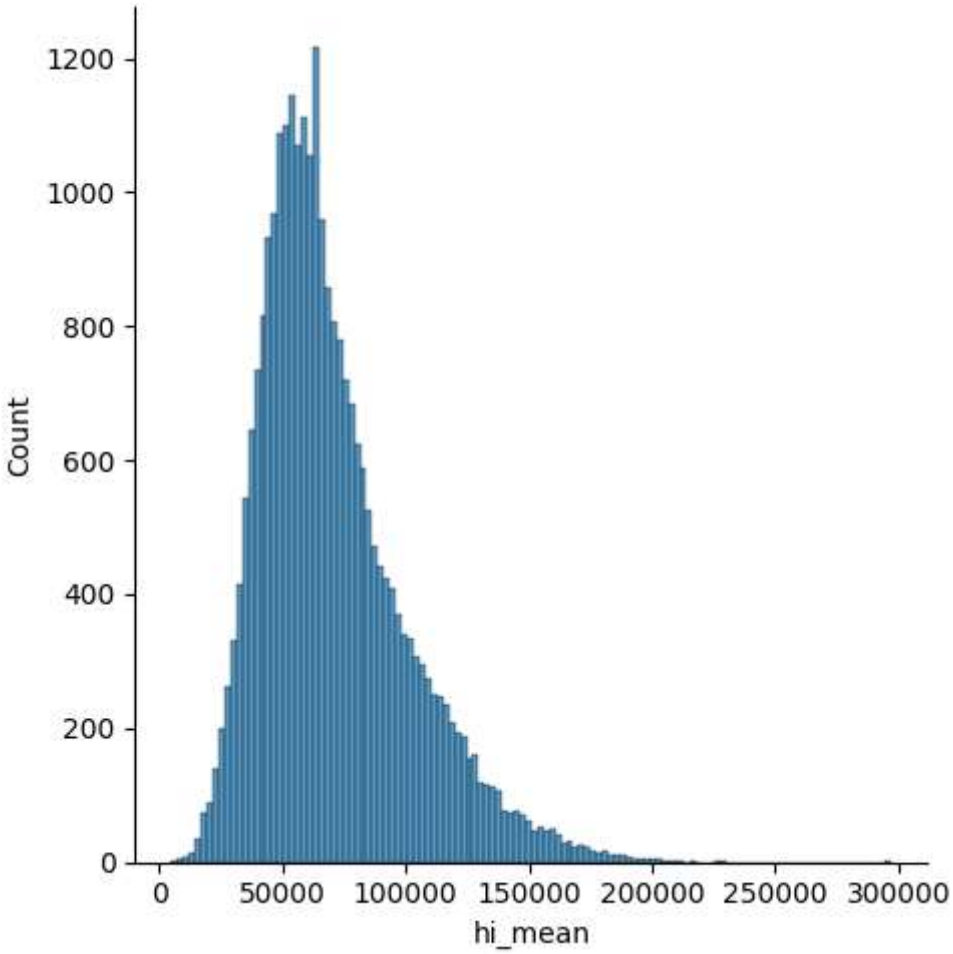
```
<seaborn.axisgrid.FacetGrid at 0x7fc05a51bbb0>
```



The family income data is slightly right skewed and also highest mean income of family is between 50000 and 100000

```
#Creating DISPLOT for HOUSEHOLD INCOME

sns.displot(df_train, x="hi_mean")
```

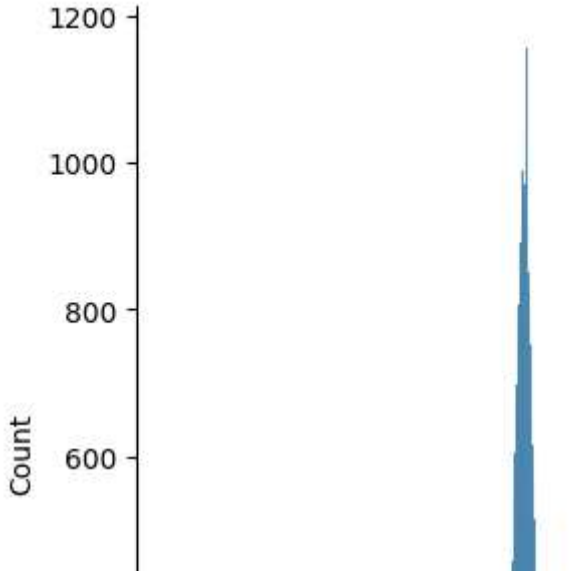
```
<seaborn.axisgrid.FacetGrid at 0x7fc05a2ccfd0>
```



The household income data is slightly right skewed and also highest mean income of household is between 50000 and 100000.

```
#Creating DISPLOT for REMAINING INCOME

sns.displot(df_train, x=df_train['family_mean']-df_train['hi_mean'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc05a1a67f0>
```



The remaining income data is right skewed and also highest mean remaining income in range of 50000 and 100000

WEEK 2:

Exploratory Data Analysis (EDA):

1. Perform EDA and come out with insights into population density and age. You may have to derive new fields (make sure to weight averages for accurate measurements):

a) Use pop and ALand variables to create a new field called population density

b) Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age

c) Visualize the findings using appropriate chart type

```
# creating new column for population density using the formula PD= Total Population/Area of l
df_train['pop_dens']=df_train['pop']/df_train['ALand']
```

```
# creating new column for population density using the formula PD= Total Population/Area of l
df_test['pop_dens']=df_test['pop']/df_test['ALand']
```

```
#Using male_age_median, female_age_median, male_pop, and female_pop to create a new field cal
#formula: Median age = ((male median* male pop)+(female median * female pop))/(Male pop+femal

df_train['median_age'] = (df_train['male_age_median']*df_train['male_pop'] + df_train['female
```

```
df_train['median_age']
```

```
UID
267822    44.67
246444    34.72
245683    41.77
279653    49.88
247218    21.97
          ...
279212    40.90
277856    39.16
233000    44.09
287425    45.03
265371    31.13
Name: median_age, Length: 27161, dtype: float64
```

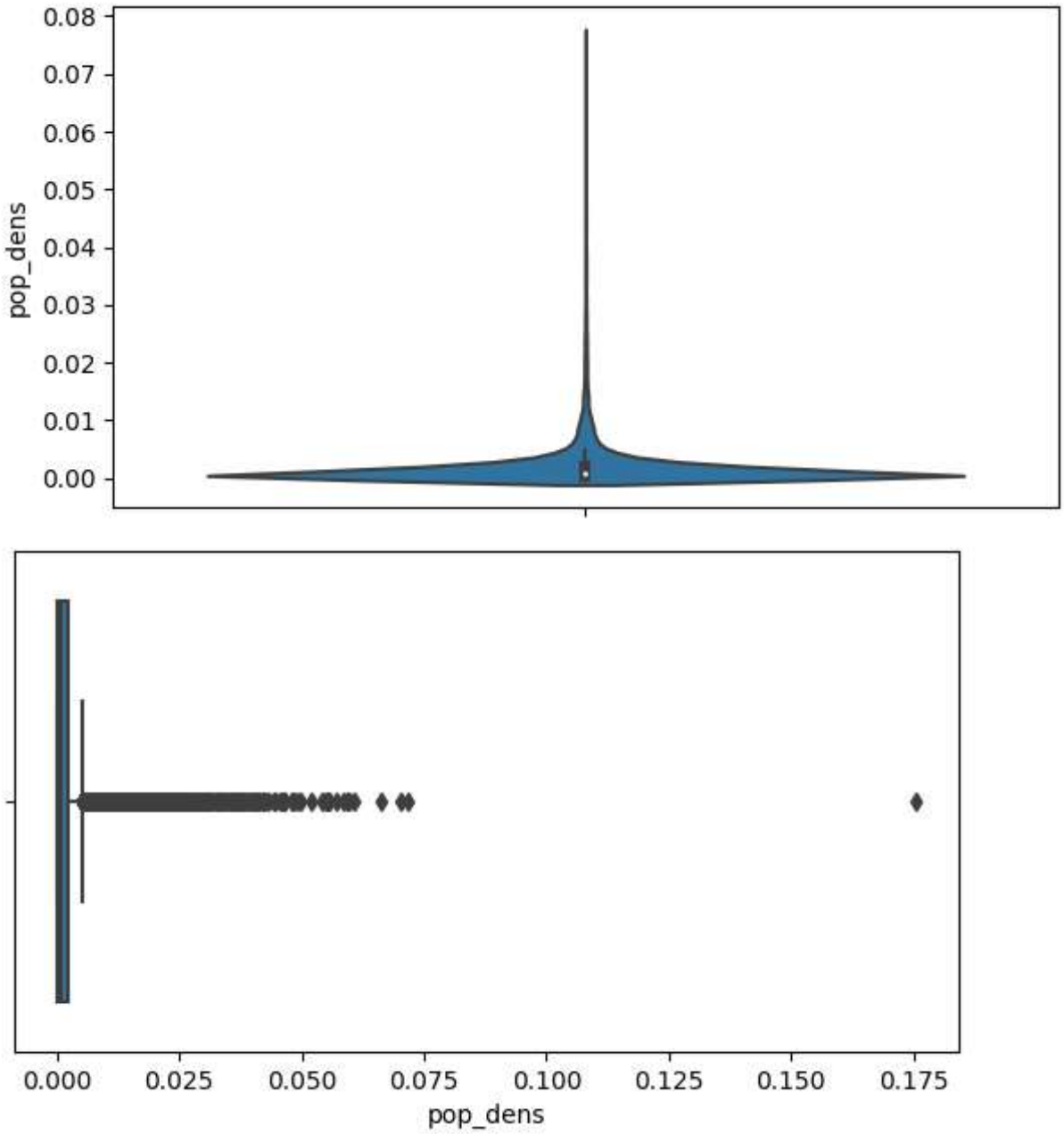
```
#Using male_age_median, female_age_median, male_pop, and female_pop to create a new field cal
#formula: Median age = ((male median* male pop)+(female median * female pop))/(Male pop+femal

df_test['median_age'] = (df_test['male_age_median']*df_test['male_pop'] + df_test['female_age
```

```
df_test['median_age']
```

```
UID
255504    31.19
252676    46.38
276314    43.15
248614    45.16
286865    43.24
          ...
238088    57.62
242811    31.16
250127    39.32
241096    44.53
287763    35.21
Name: median_age, Length: 11677, dtype: float64
```

```
# Visualize the findings using appropriate chart type
#1) Violin plot
plt.figure(figsize=(15,8))
plt.subplot(2, 2, 1)
sns.violinplot(data=df_train,y='pop_dens')
plt.figure(figsize=(15,8))
plt.subplot(2, 2, 2)
```

```
sns.boxplot(data=df_test,x='pop_dens')
plt.show()
```




```
#2) Box Plot
plt.figure(figsize=(10,7))
sns.boxplot(data=df_test,x='pop_dens')
```

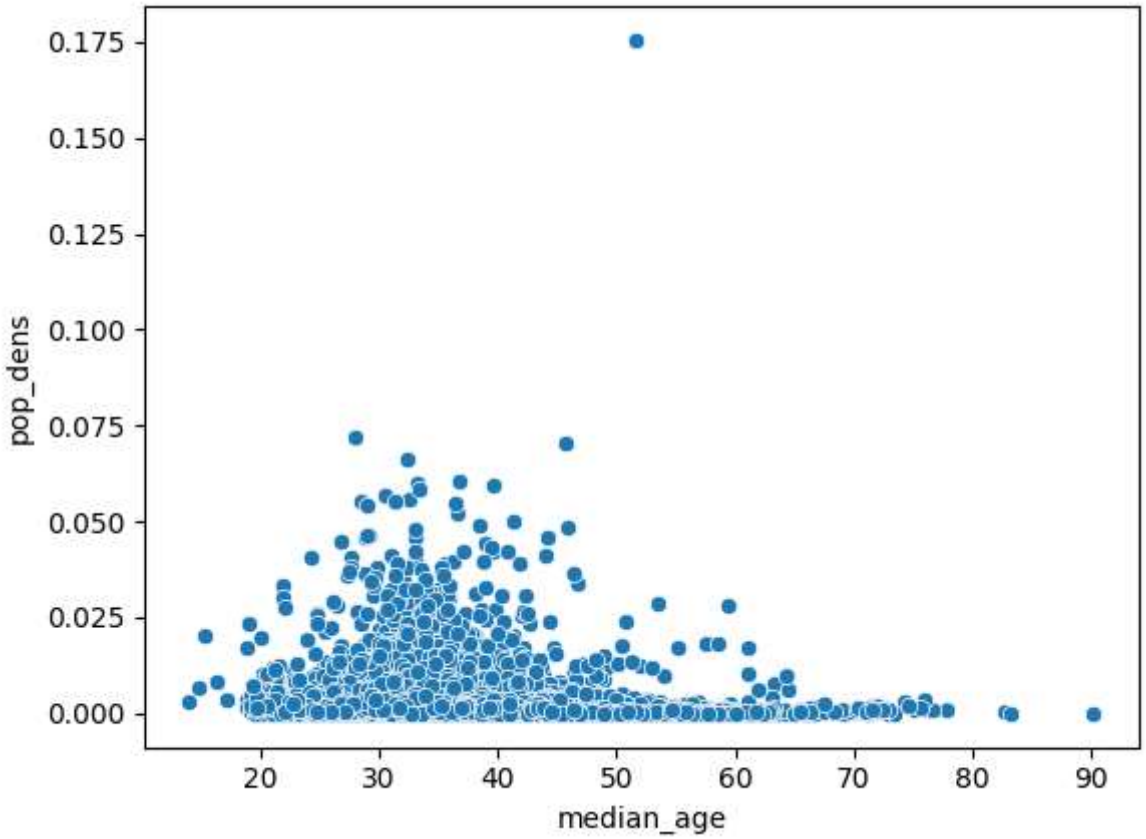
```
<Axes: xlabel='pop_dens'>
```

```
#3) Scatter plot
sns.scatterplot(data=df_train, x='median_age',y='pop_dens')
```

```
<Axes: xlabel='median_age', ylabel='pop_dens'>
```

```
#4) Scatter plot
sns.scatterplot(data=df_test, x='median_age',y='pop_dens')
```

```
<Axes: xlabel='median_age', ylabel='pop_dens'>
```



2. Create bins for population into a new variable by selecting appropriate class interval so that the number of categories don't exceed 5 for the ease of analysis.

a) Analyze the married, separated, and divorced population for these population brackets

b) Visualize using appropriate chart type

```
# create the bin for population
df_train['pop_bin']=pd.cut(df_train['pop'],bins=5,labels=['very low pop','low pop','medium pc
```

```
# count the no.of rows in population
df_train['pop'].count()
```

```
27161
```

```
df_test['pop'].count()
```

```
11677
```

```
df_train['pop_bin'].value_counts()
```

```
very low pop     26901
low pop            246
medium pop           9
high pop             4
very high pop        1
Name: pop_bin, dtype: int64
```

```
df_test['pop_bin']=pd.cut(df_test['pop'],bins=5,labels=['very low pop','low pop','medium pop'
```

```
df_test['pop_bin'].value_counts()
```

```
very low pop     11101
low pop            552
medium pop          22
high pop             1
very high pop        1
Name: pop_bin, dtype: int64
```

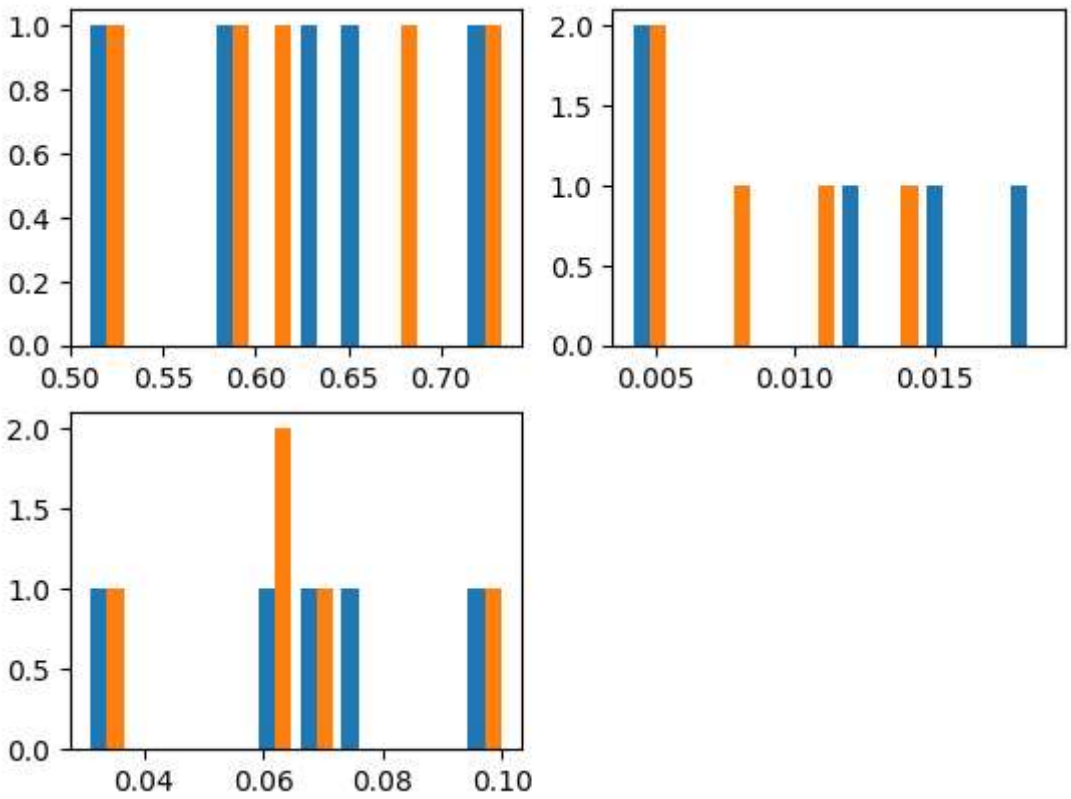
```
# count the no. of married,separated and divorced with respect to the created bin
df_train.groupby(by='pop_bin')[['married','separated','divorced']].count()
```

pop_bin	married	separated	divorced
very low pop	26901	26901	26901
low pop	246	246	246
medium pop	9	9	9
high pop	4	4	4
very high pop	1	1	1

```
# get the mean and median for married, separated and divorced with respect to the created bir
df_data=df_train.groupby(by='pop_bin')[['married','separated','divorced']].agg(['mean','media
df_data
```

	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bin						
very low pop	0.51	0.53	1.91e-02	1.35e-02	0.10	0.10
low pop	0.58	0.59	1.58e-02	1.12e-02	0.08	0.07
medium pop	0.66	0.62	5.00e-03	4.12e-03	0.07	0.06
high pop	0.63	0.68	1.24e-02	7.34e-03	0.06	0.06

```
# analyze the data using histogram,scatter plot,box and whiskers plot
plt.subplot(2,2,1)
plt.hist(data=df_data, x='married')
plt.subplot(2,2,2)
plt.hist(data=df_data, x='separated')
plt.subplot(2,2,3)
plt.hist(data=df_data, x='divorced')
plt.show()
```



3. Please detail your observations for rent as a percentage of income at an overall level, and for different states.

```
# find the overall income
df_train['overall_income']=df_train['family_mean']+df_train['hi_mean']
```

```
df_test['overall_income']=df_test['family_mean']+df_test['hi_mean']
```

```
df_train['overall_income']
```

```
UID
267822    131119.43
246444     92602.03
245683    180205.20
279653    105135.35
247218     85887.58
             ...
279212     39404.82
277856    238785.15
233000    168768.82
287425    332659.11
265371    106534.27
Name: overall_income, Length: 27161, dtype: float64
```

```
#setting the decimal to 2 places
pd.set_option('display.precision',2)
```

```
df_train['overall_income']
```

```
UID
267822    131119.43
246444     92602.03
245683    180205.20
279653    105135.35
247218     85887.58
             ...
279212     39404.82
277856    238785.15
233000    168768.82
287425    332659.11
265371    106534.27
Name: overall_income, Length: 27161, dtype: float64
```

```
# finding the rent percent using the formula
df_train['rent_percent']=df_train['rent_mean']*100/df_train['overall_income']
```

```
df_train['rent_percent']
```

```
UID
267822    0.59
246444    0.87
245683    0.41
279653    0.76
247218    1.09
          ...
```

```
279212    1.12
277856    0.76
233000    0.50
287425    0.59
265371    0.89
Name: rent_percent, Length: 27161, dtype: float64
```

```
df_test['rent_percent']=df_test['rent_mean']*100/df_test['overall_income']
```

```
df_test['rent_percent']
```

```
UID
255504    0.84
252676    0.53
276314    0.66
248614    0.51
286865    0.43
          ...
238088    1.14
242811    0.94
250127    0.57
241096    0.52
287763    0.84
Name: rent_percent, Length: 11677, dtype: float64
```

```
# take mean as aggregate so that we get rent percentage·for·individual·state
```

```
df_train.groupby(by='state')['rent_percent'].agg('mean')
```

```
state
Alabama                 0.65
Alaska                  0.69
Arizona                 0.82
Arkansas                0.63
California              0.92
Colorado                0.75
Connecticut             0.71
Delaware                0.71
District of Columbia    0.77
Florida                 0.89
Georgia                 0.75
Hawaii                  0.95
Idaho                   0.65
Illinois                0.72
Indiana                 0.69
Iowa                    0.56
Kansas                  0.62
Kentucky                0.62
Louisiana               0.72
Maine                   0.64
Maryland                0.76
Massachusetts           0.69
Michigan                0.73
```

```
Minnesota               0.61
Mississippi             0.70
Missouri                0.66
Montana                 0.59
Nebraska                0.61
Nevada                  0.83
New Hampshire           0.66
New Jersey              0.77
New Mexico              0.69
New York                0.80
North Carolina          0.69
North Dakota            0.51
Ohio                    0.66
Oklahoma                0.66
Oregon                  0.74
Pennsylvania            0.68
Puerto Rico             0.87
Rhode Island            0.69
South Carolina          0.71
South Dakota            0.51
Tennessee               0.69
Texas                   0.72
Utah                    0.71
Vermont                 0.65
Virginia                0.75
Washington              0.73
West Virginia           0.58
Wisconsin               0.65
Wyoming                 0.58
Name: rent_percent, dtype: float64
```

```
# take mean as aggregate so that we get value for individual state

df_test.groupby(by='state')['rent_percent'].agg('mean')
```

```
state
Alabama                 0.71
Alaska                  0.70
Arizona                 0.80
Arkansas                0.63
California              0.91
Colorado                0.76
Connecticut             0.76
Delaware                0.75
District of Columbia    0.86
Florida                 0.90
Georgia                 0.75
Hawaii                  1.03
Idaho                   0.65
Illinois                0.71
Indiana                 0.66
Iowa                    0.54
Kansas                  0.66
Kentucky                0.65
Louisiana               0.71
```

```
Maine                0.63
Maryland             0.77
Massachusetts        0.72
Michigan             0.74
Minnesota            0.62
Mississippi          0.71
Missouri             0.66
Montana              0.55
Nebraska             0.58
Nevada               0.86
New Hampshire        0.65
New Jersey           0.81
New Mexico           0.72
New York             0.81
North Carolina       0.68
North Dakota         0.49
Ohio                 0.67
Oklahoma             0.66
Oregon               0.75
Pennsylvania         0.68
Puerto Rico          0.88
Rhode Island         0.68
South Carolina       0.70
South Dakota         0.51
Tennessee            0.68
Texas                0.73
Utah                 0.72
Vermont              0.66
Virginia             0.76
Washington           0.71
West Virginia        0.60
Wisconsin            0.65
Wyoming              0.62
Name: rent_percent, dtype: float64
```

4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings

```
# correlation between the variables
df_train.corr()
```

	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand
SUMLEVEL	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
COUNTYID	NaN	1.00	0.22	3.58e-02	6.73e-02	-1.50e-01	7.09e-02	0.02
STATEID	NaN	0.22	1.00	-2.61e-01	4.32e-02	1.11e-01	3.19e-01	-0.02
zip_code	NaN	0.04	-0.26	1.00e+00	-5.21e-03	-7.00e-02	-9.27e-01	0.07
area_code	NaN	0.07	0.04	-5.21e-03	1.00e+00	-1.26e-01	-1.32e-02	0.02
...	...	...	...	...	...	...	...	...
Bad_Debt	NaN	-0.13	-0.15	-7.12e-02	-3.37e-03	2.11e-01	-4.43e-03	-0.08
pop_dens	NaN	-0.08	-0.02	-1.18e-01	-2.98e-02	5.29e-02	6.47e-02	-0.05
median_age	NaN	-0.06	-0.02	-1.29e-01	-1.65e-02	7.87e-03	1.08e-01	0.04

```
df_test.corr()
```

	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand
SUMLEVEL	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
COUNTYID	NaN	1.00	2.24e-01	3.14e-02	5.72e-02	-0.15	7.21e-02	-3.18e-03
STATEID	NaN	0.22	1.00e+00	-2.74e-01	3.81e-02	0.09	3.28e-01	-1.24e-02

```
df_train.columns
```

```
Index(['SUMLEVEL', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place',
       'type', 'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand',
       'AWater', 'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
       'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_gt_10',
       'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_samples',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weight',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
       'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
       'hs_degree_male', 'hs_degree_female', 'male_age_mean',
       'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
       'male_age_samples', 'female_age_mean', 'female_age_median',
       'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'Bad_Debt', 'pop_dens', 'median_age', 'pop_bin', 'overall_income',
       'rent_percent'],
      dtype='object')
```

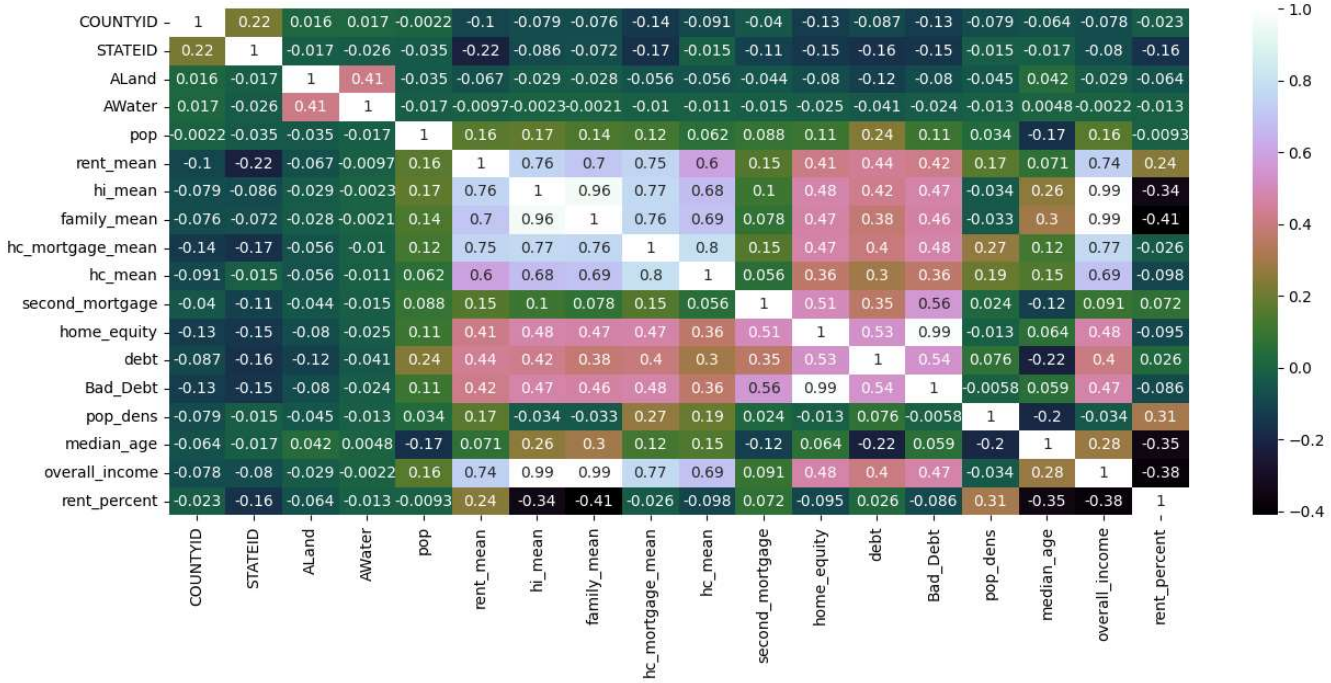
```
df_test.columns
```

```
Index(['SUMLEVEL', 'COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place',
       'type', 'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand',
       'AWater', 'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
       'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_gt_10',
       'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_samples',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weight',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
       'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
       'hs_degree_male', 'hs_degree_female', 'male_age_mean',
       'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
       'male_age_samples', 'female_age_mean', 'female_age_median',
       'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'Bad_Debt', 'pop_dens', 'median_age', 'pop_bin', 'overall_income',
```

```
            'rent_percent'],
           dtype='object')
```

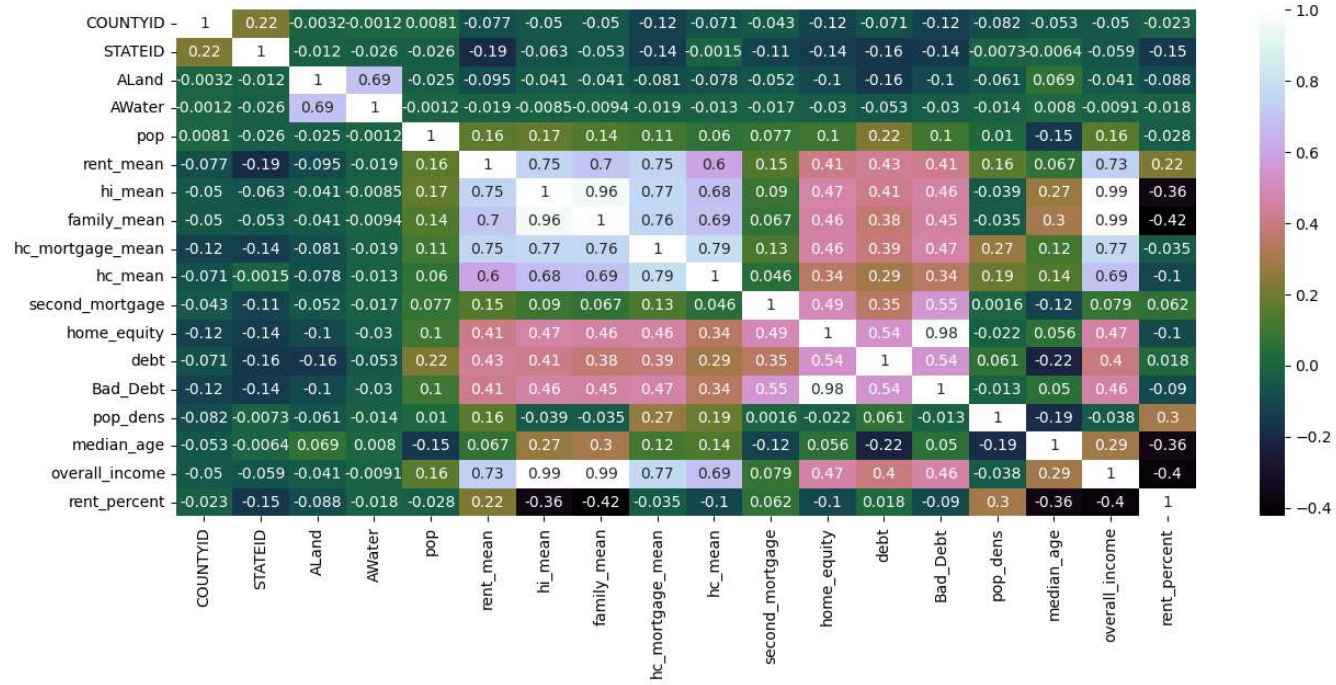
```
# selecting relevant features to display correlation
df_train_corr= df_train[['COUNTYID', 'STATEID','ALand',
       'AWater', 'pop','rent_mean','hi_mean','family_mean','hc_mortgage_mean','hc_mean','secc
       'rent_percent']]


df_test_corr= df_test[['COUNTYID', 'STATEID','ALand',
       'AWater', 'pop','rent_mean','hi_mean','family_mean','hc_mortgage_mean','hc_mean','secc
       'rent_percent']]


plt.figure(figsize=(15,6))
sns.heatmap(df_train_corr.corr(),cmap='cubehelix',annot=True)
```

```
<Axes: >
```




```
plt.figure(figsize=(15,6))
sns.heatmap(df_test_corr.corr(),cmap='cubehelix',annot=True)
```

```
<Axes: >
```



Project Task: Week 3

Data Pre-processing:

1. The economic multivariate data has a significant number of measured variables. The goal is to find where the measured variables depend on a number of smaller unobserved common factors or latent variables. 2. Each variable is assumed to be dependent upon a linear combination of the common factors, and the coefficients are known as loadings. Each measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable. Obtain the common factors and then plot the loadings. Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships in the data. Following are the list of latent variables:

- Highschool graduation rates
- Median population age
- Second mortgage statistics
- Percent own
- Bad debt expense

#

```
# Pip install factor analyser for factor analysis

!pip install factor_analyzer
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: factor_analyzer in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from fa
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from fa
Requirement already satisfied: pre-commit in /usr/local/lib/python3.9/dist-packages (fr
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from f
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: cfgv>=2.0.0 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: nodeenv>=0.11.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: virtualenv>=20.10.0 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: identify>=1.0.0 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (fr
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from
Requirement already satisfied: distlib<1,>=0.3.6 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: platformdirs<4,>=2.4 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: filelock<4,>=3.4.1 in /usr/local/lib/python3.9/dist-pack
```

```
from factor_analyzer import FactorAnalyzer
from sklearn.preprocessing import StandardScaler


# Drop the column SUMLEVEL as it holds no significance

df_train= df_train.drop(columns='SUMLEVEL',axis=1)
df_train.head()
```

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_c
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13
246444	141	18	Indiana	IN	South Bend	Roseland	City	tract	46
245683	63	18	Indiana	IN	Danville	Danville	City	tract	46
279653	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	
247218	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66

```
df_test= df_test.drop(columns='SUMLEVEL')
df_test.head()
```

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary
255504	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract
252676	1	23	Maine	ME	Auburn	Auburn City	City	tract
276314	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tract
248614	231	21	Kentucky	KY	Monticello	Monticello City	City	tract
286865	355	48	Texas	TX	Corpus Christi	Edroy	Town	tract

5 rows × 83 columns

```
df_train.columns
```

```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater',
       'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
       'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_gt_10',
       'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
```

```
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_samples',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weight',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
       'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
       'hs_degree_male', 'hs_degree_female', 'male_age_mean',
       'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
       'male_age_samples', 'female_age_mean', 'female_age_median',
       'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'Bad_Debt', 'pop_dens', 'median_age', 'pop_bin', 'overall_income',
       'rent_percent'],
      dtype='object')
```

```
# we have prior knowledge that certain variables in our dataset are highly related to the lat
# In order to focus on the remaining measured variables and identify the underlying structure
# we removed the columns corresponding to the highly related variables using the drop() metho
#This allowed us to better isolate the effects of the latent variables and identify their rel

df_train_preproc = df_train.drop(['hs_degree','median_age', 'second_mortgage', 'pct_own', 'Ba


df_test_preproc = df_test.drop(['hs_degree','median_age', 'second_mortgage', 'pct_own', 'Bad_


df_train_preproc=df_train_preproc.select_dtypes(include=('int','float'))


df_test_preproc=df_test_preproc.select_dtypes(include=('int','float'))


#check for missing values
df_train_preproc.isnull().sum().any()
```

```
False
```

```
#check for missing values
df_test_preproc.isnull().sum().any()
```

```
False
```

```
# Scale the data to have zero mean and unit variance
scaler = StandardScaler()
df_train_preproc_scaled = scaler.fit_transform(df_train_preproc)


# Scale the data to have zero mean and unit variance
scaler = StandardScaler()
df_test_preproc_scaled = scaler.fit_transform(df_test_preproc)
```

```
#Use the EFA (Exploratory factor analysis) method for factor analysis
n_factors=5
fa_train = FactorAnalyzer(n_factors,rotation ='varimax',method='principal')
```

```
#Use the EFA (Exploratory factor analysis) method for factor analysis
n_factors=5
fa_test = FactorAnalyzer(n_factors,rotation ='varimax',method='principal')
```

```
#Fitting train dataset in the factor analyzer
fa_train.fit(df_train_preproc_scaled)
```

```
FactorAnalyzer(method='principal', n_factors=5, rotation='varimax',
               rotation_kwargs={})
```

```
#Fitting test dataset in the factor analyzer
fa_test.fit(df_test_preproc_scaled)
```

```
FactorAnalyzer(method='principal', n_factors=5, rotation='varimax',
               rotation_kwargs={})
```

```
#interpret the train results
factors_train = fa_train.loadings_
factors_train
```

```
array([[-1.37067805e-01,  1.44480177e-02,  8.05486434e-03,
        -6.23338364e-02,  1.65342817e-02],
       [-1.47147181e-01,  5.80908792e-05,  1.22399083e-01,
        -1.80660711e-01,  3.12836310e-02],
       [-6.59694621e-02,  6.45810747e-02, -2.33376804e-01,
        -9.91306557e-03, -5.12702190e-02],
       [ 1.96378424e-02,  2.10713212e-02, -1.28817745e-02,
         4.18405952e-02, -2.10346121e-02],
       [ 1.58218816e-01, -8.91359132e-02, -1.18122629e-01,
        -1.64704049e-01, -5.19474689e-02],
       [-3.04174012e-02, -5.42702481e-02,  2.43191737e-01,
        -3.67935790e-02,  2.75308206e-02],
       [-6.19430173e-02, -3.16097129e-02,  6.42477349e-02,
        -1.13424756e-01, -2.20902992e-02],
       [-1.21538356e-02, -2.06589930e-02,  1.09894846e-02,
        -5.02661860e-02,  1.71924200e-03],
       [ 1.17801191e-01,  9.59367564e-01, -1.26618320e-01,
         4.10892897e-02,  8.03699326e-02],
       [ 1.10408551e-01,  9.30312963e-01, -1.44928402e-01,
         2.46413865e-02,  7.72100944e-02],
       [ 1.20779229e-01,  9.52908230e-01, -1.03932719e-01,
```

```
         5.58021851e-02,  8.05444165e-02],
       [ 8.17387077e-01,  3.54369189e-02, -1.75090459e-01,
         1.71164700e-01, -6.31225782e-02],
       [ 7.73666915e-01,  3.20696373e-02, -1.88263291e-01,
         1.70072276e-01, -6.76285778e-02],
       [ 6.87935034e-01,  2.48440393e-02,  1.08824885e-02,
         1.28766737e-01,  9.50866429e-02],
       [-3.34671687e-01,  2.93928570e-01, -7.60463826e-02,
        -3.40097965e-03,  6.82043811e-01],
       [ 1.92692627e-03,  3.80934367e-01, -1.89421278e-01,
         6.02376229e-02,  8.04273532e-01],
       [-7.97137448e-03,  5.72969825e-02, -1.19083887e-01,
         4.86738431e-01, -3.58161941e-02],
       [-7.28812012e-03,  3.77000810e-02, -1.11049635e-01,
         6.99450951e-01,  2.07101804e-02],
       [-5.90282986e-02, -5.25494160e-03, -4.87441696e-02,
         8.18913841e-01,  8.20052293e-02],
       [-9.65170060e-02, -3.01547267e-02, -5.32756659e-03,
         8.75197186e-01,  1.12177412e-01],
       [-1.15078036e-01, -4.16724849e-02,  2.06815950e-02,
         8.91585955e-01,  1.10291487e-01],
       [-1.07494581e-01, -5.26829400e-02,  3.67800052e-02,
         8.79584293e-01,  1.10284174e-01],
       [-1.05994966e-01, -6.08018588e-02,  4.04190049e-02,
         8.49507634e-01,  1.13254454e-01],
       [-9.23601013e-02, -7.76809098e-02,  4.03557357e-02,
         7.70850171e-01,  1.24834640e-01],
       [-2.00321996e-02,  4.05070555e-01, -1.60478827e-01,
         5.62893169e-02,  8.02473886e-01],
       [ 9.42199862e-03,  3.89365109e-01, -1.84451041e-01,
         5.55640597e-02,  7.97773077e-01],
       [ 8.94478616e-01,  8.75632204e-02, -3.59945489e-02,
        -1.99715314e-01, -2.82926438e-01],
       [ 8.47272767e-01,  9.41402416e-02, -8.09142850e-02,
        -2.05476579e-01, -3.20492971e-01],
       [ 8.91101749e-01,  5.87773572e-02,  1.04001090e-01,
```

```
#interpret the train results
factors_test = fa_test.loadings_
factors_test
```

```
array([[-0.10888924,  0.01937247,  0.01420509, -0.07252267,  0.01131823],
       [-0.11371582,  0.00297329,  0.14540021, -0.18546459,  0.05171111],
       [-0.06305966,  0.04653721, -0.28366159, -0.0234948 , -0.07731624],
       [ 0.0281347 ,  0.01647944, -0.00488141,  0.04974946, -0.01961533],
       [ 0.17940702, -0.0797518 , -0.11358367, -0.15371745, -0.05567248],
       [-0.02095446, -0.03712932,  0.29339464, -0.02574904,  0.05751099],
       [-0.08678423, -0.0178266 ,  0.09973177, -0.16295351, -0.03467636],
       [-0.02277998, -0.00969135,  0.02782143, -0.07401846,  0.01662171],
       [ 0.11590341,  0.9599222 , -0.10605978,  0.02685158,  0.07498189],
       [ 0.10554961,  0.92299097, -0.12438414,  0.00655292,  0.07504154],
       [ 0.12081528,  0.95174262, -0.0827327 ,  0.04590429,  0.0713961 ],
       [ 0.81792342,  0.0363229 , -0.18601137,  0.17226956, -0.04854071],
       [ 0.77485165,  0.03275065, -0.19351602,  0.16949175, -0.05127236],
       [ 0.69476818,  0.02985963, -0.00267449,  0.14059784,  0.07708043],
```

```
[-0.32989419,  0.31415043, -0.03439165, -0.00598326,  0.67570488],
[ 0.01190232,  0.40103539, -0.13739549,  0.05803836,  0.7943229 ],
[-0.02207137,  0.06009799, -0.10902889,  0.49146801, -0.02437385],
[-0.03564451,  0.04114647, -0.10011218,  0.69407672,  0.04644313],
[-0.07765214,  0.00426697, -0.04018618,  0.81871545,  0.09280618],
[-0.12433699, -0.02046035,  0.00815302,  0.86644079,  0.13578903],
[-0.12063579, -0.02555179,  0.02765256,  0.88569834,  0.13811258],
[-0.12127907, -0.03498787,  0.0440778 ,  0.87533452,  0.13809323],
[-0.11333066, -0.04690975,  0.0525278 ,  0.84333899,  0.14345597],
[-0.10286028, -0.05419986,  0.05198579,  0.7683628 ,  0.15326231],
[-0.01012084,  0.42520578, -0.10922391,  0.05366542,  0.78908354],
[ 0.02015591,  0.40931624, -0.13298533,  0.05282115,  0.78543849],
[ 0.8948118 ,  0.0810927 , -0.05221141, -0.19961668, -0.28705755],
[ 0.84976662,  0.08171839, -0.09200142, -0.20465151, -0.3153945 ],
[ 0.8889024 ,  0.06347615,  0.08055399, -0.15010628, -0.15945115],
[-0.30764668,  0.82026658,  0.14300791,  0.05599942,  0.29191751],
[ 0.14386   ,  0.94052637,  0.07502517, -0.04666217,  0.1128192 ],
[ 0.90463907,  0.0621768 ,  0.01009702, -0.2125006 , -0.2417783 ],
[ 0.87821968,  0.05381625, -0.0048497 , -0.21359135, -0.24620381],
[ 0.82872226,  0.06337431,  0.09079917, -0.13878116, -0.10991278],
[-0.34015429,  0.85408851,  0.05063089,  0.08772226,  0.07077744],
[ 0.14792734,  0.94129716,  0.00850938, -0.02675889, -0.14918174],
[ 0.92303063, -0.02862215, -0.04716529,  0.07611448,  0.12190673],
[ 0.9093168 , -0.03715555, -0.05728808,  0.08258239,  0.12436366],
[ 0.78126086,  0.00701099,  0.10278207,  0.02479336,  0.02743174],
[-0.2171158 ,  0.74356699,  0.01841148, -0.11255043, -0.43286512],
[ 0.29616516,  0.76153064, -0.09233987, -0.05752159, -0.43792368],
[ 0.84816205, -0.05343041,  0.07750278,  0.00964797,  0.15498653],
[ 0.81701965, -0.05645175,  0.06744851,  0.01511861,  0.16183247],
[ 0.67576991, -0.0098878 ,  0.21625543, -0.01115307,  0.14961746],
[-0.10988002,  0.58958556,  0.56420198, -0.12656724, -0.31264778],
[-0.32831909,  0.54138348,  0.51844502, -0.12932362, -0.30751719],
[ 0.07705906,  0.0641505 , -0.45643615,  0.113448  , -0.20427704],
[ 0.5032118 ,  0.05838098, -0.38240973,  0.08410755, -0.26950866],
[ 0.42247085,  0.15382398, -0.64940484,  0.07661109, -0.14377412],
[-0.1639492 , -0.15004837,  0.37720716, -0.07435206,  0.38027029],
[-0.53081277, -0.09114082,  0.36189206, -0.07529811,  0.32432632],
[-0.43012175, -0.1486337 ,  0.68093352, -0.08898525,  0.12170693],
[ 0.53169765,  0.09703733,  0.02178323, -0.23861495, -0.30578162],
[ 0.48885662,  0.0922918 ,  0.0484309 , -0.25780136, -0.35221301],
[ 0.20004142, -0.06704654,  0.72817636, -0.08992974, -0.36257085],
[ 0.24914385, -0.05072875,  0.6713895 , -0.10888343, -0.43448901],
[-0.01259603,  0.07565604,  0.50305161, -0.03155574, -0.38041431],
[ 0.0660861 ,  0.84352226, -0.17489708,  0.03657774,  0.12335135]
```

Project Task: Week 4

Data Modeling :

1. Build a linear Regression model to predict the total monthly expenditure for home mortgages loan. Please refer 'deplotment_RE.xlsx'. Column hc_mortgage_mean is predicted variable. This is the mean monthly mortgage and owner costs of specified geographical location.

Note: Exclude loans from prediction model which have NaN (Not a Number) values for hc_mortgage_mean.

a) Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step.

b) Run another model at State level. There are 52 states in USA.

c) Keep below considerations while building a linear regression model. Data Modeling :

• Variables should have significant impact on predicting Monthly mortgage and owner costs

• Utilize all predictor variable to start with initial hypothesis

• R square of 60 percent and above should be achieved

• Ensure Multi-collinearity does not exist in dependent variables

• Test if predicted variable is normally distributed

```
# Select train columns with categorical and object datatypes

df_cat_train=df_train.select_dtypes(include=('object','category')).columns


# Select test columns with categorical and object datatypes

df_cat_test=df_test.select_dtypes(include=('object','category')).columns


df_cat_train

     Index(['state', 'state_ab', 'city', 'place', 'type', 'primary', 'pop_bin'],
     dtype='object')


df_cat_test

     Index(['state', 'state_ab', 'city', 'place', 'type', 'primary', 'pop_bin'],
     dtype='object')


# label encoding for the categorical datatypes

from sklearn.preprocessing import LabelEncoder
for df_cat_train in df_train:
    encoder_train = LabelEncoder()
    df_train[df_cat_train] = encoder_train.fit_transform(df_train[df_cat_train])



# label encoding for the categorical datatypes

from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.preprocessing import LabelEncoder
for df_cat_test in df_test:
    encoder_test = LabelEncoder()
    df_test[df_cat_test] = encoder_test.fit_transform(df_test[df_cat_test])
```

```
df_train.isnull().sum().any()
```

```
False
```

```
df_test.isnull().sum().any()
```

```
False
```

```
df_train
```

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_c
267822	32	32	32	34	2520	3611	2	0	1638	
246444	84	14	14	15	5758	7584	2	0	6266	
245683	38	14	14	15	1476	2095	2	0	6159	
279653	76	51	39	39	5428	3554	4	0	87	
247218	96	16	16	16	3625	5248	2	0	8679	
...	...	...	...	...	...	...	...	...	...	
279212	26	51	39	39	1197	1725	4	0	60	
277856	56	38	38	38	541	781	0	0	2357	
233000	53	5	5	5	6556	7667	2	0	10525	
287425	231	43	44	44	1234	1772	3	0	9722	
265371	1	28	28	33	3300	6698	2	0	11148	

27161 rows × 83 columns

```
df_test.head()
```

UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_c
255504	97	22	22	22	950	1216	1	0	4049	
252676	0	19	19	21	160	201	2	0	314	
276314	9	38	38	38	3017	3295	0	0	1123	
248614	132	17	17	17	2486	3374	2	0	3480	
286865	180	43	44	44	808	1472	3	0	6114	

```
# separating feature columns from target variables

feature_train = df_train.drop('hc_mortgage_mean',axis=1)
```

```
feature_test = df_test.drop('hc_mortgage_mean',axis=1)


target_train = df_train['hc_mortgage_mean']


target_test = df_test['hc_mortgage_mean']


# Calculate correlation coefficient between 'feature' and 'target'
correlations_train = feature_train.corrwith(target_train)


correlations_test = feature_test.corrwith(target_test)


# sort the correlations in descending order
correlations_train = correlations_train.sort_values(ascending=False)


correlations_test = correlations_test.sort_values(ascending=False)


correlations_train
```

```
hc_mortgage_median    0.93
hc_mortgage_stdev     0.83
hc_mean               0.79
rent_mean             0.77
hi_stdev              0.77
                      ...
hc_sample_weight     -0.39
divorced             -0.44
debt_cdf             -0.49
home_equity_cdf      -0.54
primary                NaN
Length: 82, dtype: float64
```

```
correlations_test
```

```
hc_mortgage_median    0.94
hc_mortgage_stdev     0.83
hc_mean               0.78
rent_mean             0.78
hi_stdev              0.76
                      ...
hc_sample_weight     -0.38
divorced             -0.44
debt_cdf             -0.48
home_equity_cdf      -0.54
primary                NaN
Length: 82, dtype: float64
```

```
# select the features with correlation between 0.6 to 1 or between -0.6 to -1
selected_features_train = correlations_train.loc[(correlations_train >= 0.5) & (correlations_
```

```
# select the features with correlation between 0.6 to 1 or between -0.6 to -1
selected_features_test = correlations_test.loc[(correlations_test >= 0.5) & (correlations_tes
```

```
selected_features_train
```

```
Index(['hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mean', 'rent_mean',
       'hi_stdev', 'hi_mean', 'overall_income', 'family_mean', 'family_stdev',
       'hc_median', 'hi_median', 'family_median', 'rent_median', 'rent_stdev',
       'hc_stdev', 'Bad_Debt', 'home_equity', 'home_equity_cdf'],
      dtype='object')
```

```
selected_features_test
```

```
Index(['hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mean', 'rent_mean',
       'hi_stdev', 'hi_mean', 'overall_income', 'family_mean', 'family_stdev',
       'hc_median', 'rent_median', 'hi_median', 'family_median', 'rent_stdev',
       'hc_stdev', 'Bad_Debt', 'home_equity', 'home_equity_cdf'],
      dtype='object')
```

```
# Predicting on the basis of mortgage mean

X = df_train[selected_features_train]
y = df_train['hc_mortgage_mean']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=23)
```

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# model predicting the target variable
y_pred = lin_reg.predict(X_test)
```

```
#difference between the actual and the predicted
residual_train_nation = y_test-y_pred
residual_train_nation
```

```
UID
284189    2462.23
292438     356.18
266974    1487.26
224472   -2460.77
242920     549.74
           ...
287032   -1186.20
271658     530.27
225231     323.30
276401   -1716.57
258548    5615.81
Name: hc_mortgage_mean, Length: 8149, dtype: float64
```

```
# check the RMSE value
from sklearn.metrics import mean_squared_error, r2_score
rmse_model_1 = np.sqrt(mean_squared_error(y_test, y_pred))
# Check the R2 score
r2_model_1 = r2_score(y_test, y_pred)
print('The RMSE value for model_1 is:',rmse_model_1)
print('The R2 score for Model_2 is:',r2_model_1)
```

```
The RMSE value for model_1 is: 1770.9651857783201
The R2 score for Model_2 is: 0.9454155797032244
```

```
# Using test data to predict the target
# Note-The column sequence for test data should match the model sequence
Test_nation=df_test[['hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mean', 'rent_mean',
       'hi_stdev', 'hi_mean', 'overall_income', 'family_mean', 'family_stdev',
       'hc_median', 'hi_median', 'family_median', 'rent_median', 'rent_stdev',
       'hc_stdev', 'Bad_Debt', 'home_equity', 'home_equity_cdf']]
```

```
# predict the value
predict_nation=lin_reg.predict(Test_nation)
```

```
predict_nation
```

```
array([10013.47369449, 12939.90691668, 11081.00199706, ...,
       15809.34433803, 10966.63423804, 12350.84744147])
```

```
# Check the residual
residual_test_nation=predict_nation-df_test['hc_mortgage_mean']
```

```
residual_test_nation
```

```
UID
255504    7330.47
252676    6775.91
276314    7184.00
248614    9072.21
286865    8245.87
           ...
238088    6890.24
242811    6769.54
250127    8173.34
241096    7822.63
287763    7446.85
Name: hc_mortgage_mean, Length: 11677, dtype: float64
```

The residual is too high we should run another model at state level.

Run another model at State level. There are 52 states in USA.

```
# Selecting few states to make another model
df_train['STATEID'].value_counts()
```

```
4     2905
43    1928
32    1769
9     1594
38    1219
13    1110
35    1087
22    1037
33     824
10     770
30     696
46     690
14     572
21     549
47     542
2      537
49     536
20     529
```

```
42     527
23     499
25     495
5      477
18     428
0      412
17     399
40     394
36     369
6      325
37     322
51     313
16     301
15     288
3      268
24     244
28     243
44     230
31     197
27     188
48     168
19     130
11     125
26     120
29     112
12     101
39      90
41      89
34      80
1       75
7       71
8       66
45      63
50      58
Name: STATEID, dtype: int64
```

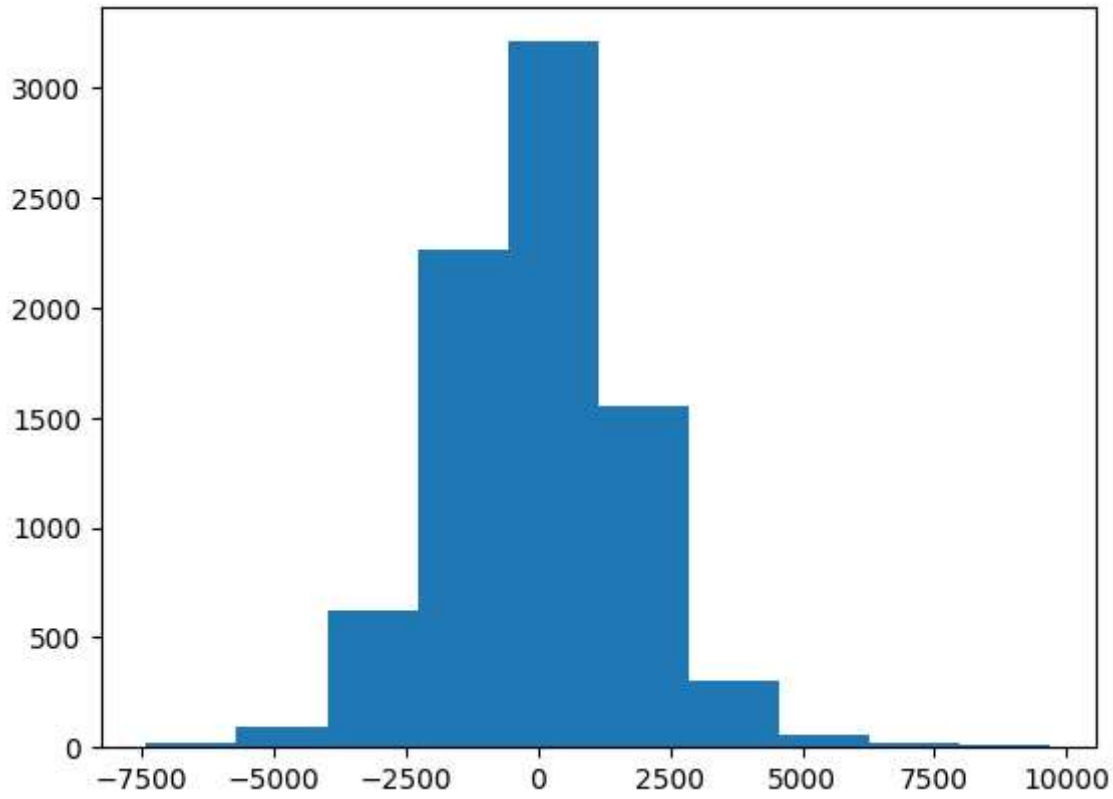
```
# Select three state ID's for further process
state_id=(4,43,32)


# run a for loop and build a second model based on stateid
for i in state_id:
  x_state=df_train[df_train['STATEID']==i][selected_features_train]
  y_state=df_train[df_train['STATEID']==i]['hc_mortgage_mean']

  x_state_train,x_state_test,y_state_train,y_state_test = train_test_split(x_state,y_state,te
  lin_reg_state = LinearRegression()
  lin_reg_state.fit(x_state_train,y_state_train)
  Predict_state = lin_reg_state.predict(x_state_test)

  rmse_model_2 = np.sqrt(mean_squared_error(y_state_test,Predict_state))
  r2_model_2 = r2_score(y_state_test,Predict_state)

  print('State ID =',i)
```

```
print('The RMSE value for the Model_2 is =',rmse_model_2)
print('The R2 score for the Model_2 is =',r2_model_2)
```

```
State ID = 4
The RMSE value for the Model_2 is = 1577.5962168075764
The R2 score for the Model_2 is = 0.8969324281393487
State ID = 43
The RMSE value for the Model_2 is = 1720.3984229742382
The R2 score for the Model_2 is = 0.9327009520834112
State ID = 32
The RMSE value for the Model_2 is = 2029.7469435286969
The R2 score for the Model_2 is = 0.9294714299039993
```

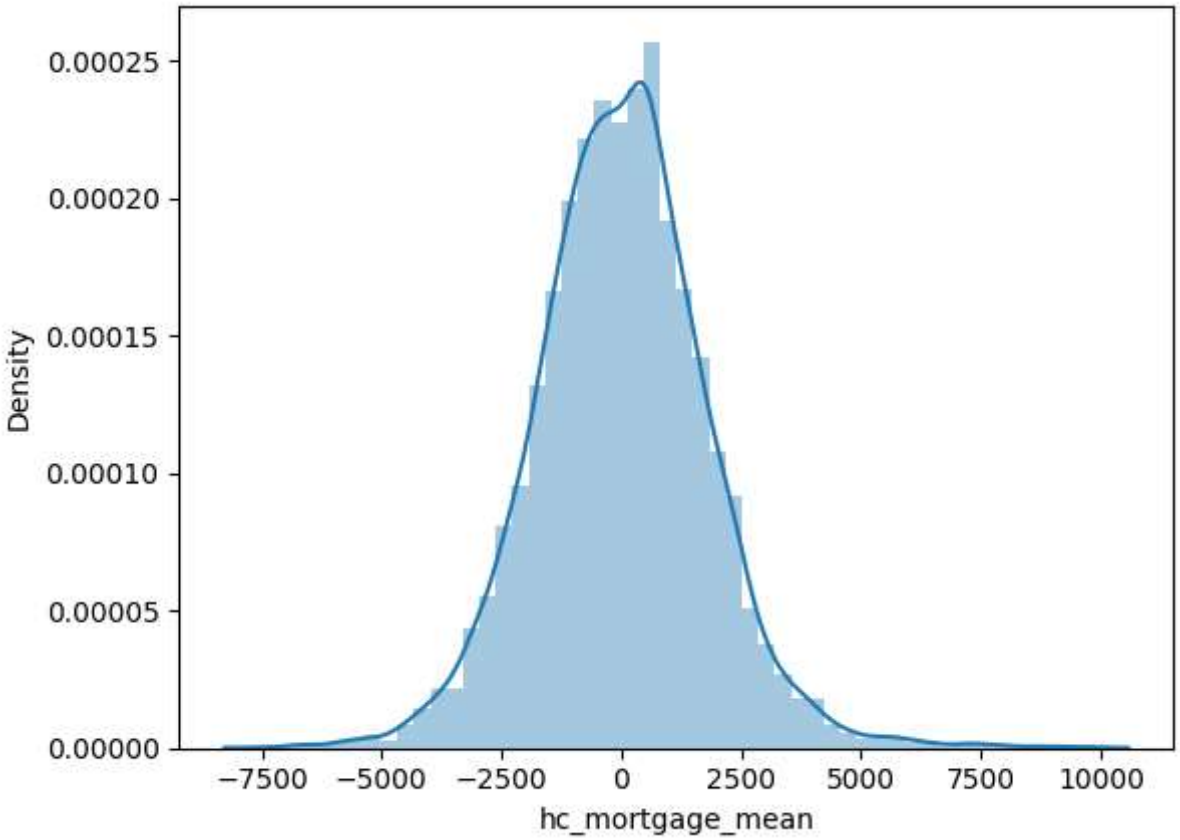
```
plt.hist(residual_train_nation)
```

```
(array([  17.,   92.,  620., 2267., 3209., 1556.,  303.,   57.,   20.,
          8.]),
 array([-7404.74816065, -5694.8545289 , -3984.96089715, -2275.0672654 ,
        -565.17363365,  1144.7199981 ,  2854.61362985,  4564.5072616 ,
        6274.40089335,  7984.2945251 ,  9694.18815685]),
 <BarContainer object of 10 artists>)
```




```
sns.distplot(residual_train_nation)
```

```
<ipython-input-931-3235b20bb07b>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(residual_train_nation)
<Axes: xlabel='hc_mortgage_mean', ylabel='Density'>
```



Colab paid products - Cancel contracts here

✓ 0s completed at 12:10 PM